Learning Balls of Strings: A Horizontal Analysis

Colin de La Higuera, Jean-Christophe Janodet, Frédéric Tantini

Learning Balls of Strings: A Horizontal Analysis*

Colin de la Higuera and Jean-Christophe Janodet and Frédéric Tantini

Laboratoire Hubert Curien, Université Jean Monnet
18 rue du Professeur Benoît Lauras, 42000 Saint-Étienne, France
{cdlh,janodet,frederic.tantini}@univ-st-etienne.fr

Abstract. There are a number of established paradigms to study the learnability of classes of functions or languages: Query learning, Identification in the limit, Probably Approximately Correct learning. Comparison between these paradigms is hard. Moreover, when to the question of converging one adds computational constraints, the picture becomes even less clear. We concentrate here on just one class of languages, that of topological balls of strings (for the edit distance), and visit the different learning paradigms in this context. Between the results, we show that surprisingly it is technically easier to learn from text than from an informant.
Keywords: Language learning, Query learning, Polynomial identification in the limit, PAC-learning.

1 Introduction

When aiming to prove that a class of languages is learnable there have been typically three different settings:

- Identification in the limit [1, 2] sees the learning process as one where information keeps on arriving about a target language. The Learner keeps making new hypothesis. Convergence takes place if there is always a moment where the process is stationary and then the hypothesis is correct.
- PAC-learning consists in learning in a setting where a distribution over the strings can be used to sample examples both to build the hypothesis and to test this hypothesis [3]. Two parameters can be fixed. The first (ϵ) measures the error that is made (which corresponds to the probability that a string is classified wrongly), and the second (δ) measures the probability that the sampling process has gone wrong.
- Query learning (sometimes called active learning) is the process of being able to interrogate an Oracle about the language to be learned, in a formalised way [4].

The three settings are usually difficult to compare; Using computability theory, one reference is [5]. The comparison becomes even harder when complexity

* This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

issues are discussed. Some exceptions are the work by Angluin comparing PAC-learning and using equivalence queries [6], the work by Pitt relating equivalence queries and implicit prediction errors [7], comparisons between learning with characteristic samples, simple PAC [8] and MAT in [9]. Other analysis of polynomial aspects of learning grammars, automata and languages can be found in [7, 10, 11]. An alternative approach to efficiency issues in inductive inference can be found (for pattern languages) in [12].

If the customary approach is to introduce a learning paradigm and survey a variety of classes of languages for this paradigm, we choose here to visit just one class, that of balls of strings for the edit or levenshtein distance [13]. These were shown to be learnable from noise [14] and from correction queries [15].

The results we obtain here are generally negative: Polynomial identification in the limit from an informant is impossible, for most definitions, and it is the same when learning from membership and equivalence queries. PAC-learning is also impossible in polynomial time, at least if we accept that $\mathcal{RP} \neq \mathcal{NP}$.

On the other hand, the errors are usually due to the counterexamples, and if we learn from text (instead of from an informant or when we count only mind changes) we get several positive results: Only a polynomial number of mind changes or of prediction errors are made. This constitutes a small surprise as one would think the more information one gets for learning, the richer the classes one can learn would be.

In Section 2 we introduce the definitions corresponding to balls of strings and complexity classes. In Sections 3, 4 and 5, we focus on so-called good balls only and present the results concerning PAC-learning, query learning and polynomial identification in the limit, respectively. We list our learning results on general balls in Section 6 before concluding in Section 7.

2 Definitions

2.1 Languages and Grammars

An *alphabet* Σ is a finite nonempty set of symbols called *letters*. We suppose in the sequel that $|\Sigma| \geq 2$. A *string* $w = a_1 \dots a_n$ is any finite sequence of letters. We write λ for the empty string and $|w|$ for the length of w. Let $\Sigma^\star$ denote the set of all strings over Σ. A *language* is any subset $L \subseteq \Sigma^\star$. Let $\mathbb{N}$ be the set of non negative integers. For all $k \in \mathbb{N}$, let $\Sigma^{\leq k} = \{w \in \Sigma^\star : |w| \leq k\}$ and $\Sigma^{>k} = \{w \in \Sigma^\star : |w| > k\}$. The symmetrical difference $A \oplus B$ between 2 languages A and B is the set of all strings that belong to exactly one of them.

Grammatical inference aims at learning the languages of a fixed class $\boldsymbol{\mathcal{L}}$ (semantics) represented by the grammars of a given class $\boldsymbol{\mathcal{G}}$ (syntax). $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{G}}$ are related by a semantical naming function $\mathbb{L} : \boldsymbol{\mathcal{G}} \to \boldsymbol{\mathcal{L}}$ that is total ($\forall G \in \boldsymbol{\mathcal{G}}, \mathbb{L}(G) \in \boldsymbol{\mathcal{L}}$) and surjective ($\forall L \in \boldsymbol{\mathcal{L}}, \exists G \in \boldsymbol{\mathcal{G}}$ *s.t.* $\mathbb{L}(G) = L$). For any string $w \in \Sigma^\star$ and language $L \in \boldsymbol{\mathcal{L}}$, we shall write $L \models w$ *if*$_{def}$ $w \in L$. Concerning the grammars, they may be understood as any piece of information allowing a given parser to recognize strings. For any string $w \in \Sigma^\star$ and representation $G \in \boldsymbol{\mathcal{G}}$,

we shall write $G \vdash w$ if the parser answers YES given G and w. Basically, we require that the semantical function matches the parser: $G \vdash w \iff \mathbb{L}(G) \models w$.

Finally, we will mainly consider learning paradigms subject to efficiency constraints. In order to define them, we will use $\|G\|$ to denote the size of the grammar G (*e.g.*, the number of states in the case of DFA). Moreover, given a set S of strings, we will use $\|S\|$ to denote the sum of the lengths of the strings in S. Finally, we will use the single bar notation $|\cdot|$ for the cardinality of sets.

2.2 Balls of Strings

The *edit distance* $d(w, w')$ is the minimum number of *primitive edit operations* needed to transform w into w' [13]. The primitive operation is either (1) a *deletion*: $w = uav$ and $w' = uv$, or (2) an *insertion*: $w = uv$ and $w' = uav$, or (3) a *substitution*: $w = uav$ and $w' = ubv$, where $u, v \in \Sigma^\star$, $a, b \in \Sigma$ and $a \neq b$. *E.g.*, $d(abaa, aab) = 2$ since $a\underline{b}aa \to aa\underline{a} \to aab$ and the rewriting of $abaa$ into aab cannot be achieved with less than 2 steps. Notice that $d(w, w')$ can be computed in $\mathcal{O}(|w| \cdot |w'|)$ time by dynamic programming [16].

It is well-known that the edit distance is a *metric* [17], so it conveys to $\Sigma^\star$ the structure of a *metric space*. Therefore, it is natural to introduce the balls of strings. The *ball of centre* $o \in \Sigma^\star$ *and radius* $r \in \mathbb{N}$, denoted $B_r(o)$, is the set of all strings whose distance is at most r from o: $B_r(o) = \{w \in \Sigma^\star : d(o, w) \leq r\}$. *E.g.*, if $\Sigma = \{a, b\}$, then $B_1(ba) = \{a, b, aa, ba, bb, aba, baa, bab, bba\}$ and $B_r(\lambda) = \Sigma^{\leq r}$ for all $r \in \mathbb{N}$. We denote by $\boldsymbol{\mathcal{BALL}}(\Sigma)$ the family of all the balls: $\boldsymbol{\mathcal{BALL}}(\Sigma) = \{B_r(o) : o \in \Sigma^\star, r \in \mathbb{N}\}$.

We represent a ball $B_r(o)$ by the pair (o, r) itself. Indeed, its size is $|o| + \log r$. Moreover, deciding whether $w \in B_r(o)$ or not is immediate: One only has to (1) compute $d(o, w)$ and (2) check whether this distance is at most r, which is achievable in time $\mathcal{O}(|o| \cdot |w| + \log r)$. Finally, as $|\Sigma| \geq 2$, we can show that (o, r) is a unique thus *canonical* representation of $B_r(o)$ [15].

A *good ball* is a ball whose radius is at most the length of the centre. The advantage of using good balls is that there is a natural relation between the size of the centre and the size of the border strings. We denote by $\boldsymbol{\mathcal{GB}}(\Sigma)$ the class of all the good balls.

2.3 Complexity Classes

See [18] for a comprehensive survey. Here, we only wish to recall that $\mathcal{RP}$ (*'Randomised Polynomial Time'*) is the complexity class of decision problems for which a probabilistic Turing machine exists and (1) it always runs in time polynomial in the input size, (2) if the correct answer is NO, it always returns NO and (3) if the correct answer is YES, then it returns YES with probability $> \frac{1}{2}$ (otherwise, it returns NO).

The algorithm is randomised since it is allowed to flip a random coin while it is running. It should be noted that because the error (in the negative case) is less that 0.5, by repeating the run of the algorithm as many times as necessary,

the actual error can be brought to be as small as one wants. Notice that the algorithm only makes one sided errors. The strong belief and assumption is that $\mathcal{RP} \neq \mathcal{NP}$.

3 Main Pac Results

The Pac (Probably Approximatively Correct) paradigm has been widely used in machine learning to provide a theoretical setting for convergence issues. The setting was introduced by [3], and the analysis for the case of learning from strings representations of unbounded size have always been tricky [19, 10, 20]. Typical techniques proving non Pac-learnability often depend on complexity hypothesis [21].

3.1 Balls of Strings are not Pac-Learnable

Definition 1 (ϵ-good hypothesis). *Let G be the target grammar and H be a hypothesis grammar. Let $\mathcal{D}$ be a distribution over $\Sigma^\star$. We say, for $\epsilon > 0$, that H is an* ϵ-good hypothesis *with respect to G if$_{def}$ $Pr_{\mathcal{D}}(x \in \mathbb{L}(G) \oplus \mathbb{L}(H)) < \epsilon$.*

In the usual definition of Pac-learning we are going to sample examples to learn from. In the case of strings, there always is the risk (albeit small) to sample a string too long to account for in polynomial time. In order to avoid this problem, we will sample from a distribution restricted to strings shorter than a specific value given by the following lemma:

Lemma 1. *Let $\mathcal{D}$ be a distribution over $\Sigma^\star$. Then given any $\epsilon > 0$ and any $\delta > 0$, with probability at least 1-δ we have: If we draw a sample X of size at least $\frac{1}{\epsilon} \ln \frac{1}{\delta}$ strings following $\mathcal{D}$ and write $\mu_X = \max\{|y| : y \in X\}$, then $Pr_{\mathcal{D}}(|x| > \mu_X) < \epsilon$.*

Proof. Denote by μ_ϵ the smallest integer n such that $Pr(\Sigma^{>n}) < \epsilon$. A sufficient condition for $Pr_{\mathcal{D}}(|x| > \mu_X) < \epsilon$ is that we take a sample large enough to be nearly sure (*i.e.* with probability at least $1 - \delta$) to have one string longer than μ_ϵ. On the contrary, the probability of having all (n) strings in X of length at most μ_ϵ is bounded by $(1 - \epsilon)^n$. In order for this quantity to be less than δ, it is sufficient to take $n \geq \frac{1}{\epsilon} \ln \frac{1}{\delta}$. □

A learning algorithm is now asked to learn a grammar given a *confidence* parameter δ and an *error* parameter ϵ. The algorithm must also be given an upper bound on the size of the target grammar and on the length of the examples it is going to get (perhaps using an extra sample built thanks to Lemma 1 above). The algorithm can query an Oracle: It may ask for an example randomly drawn according to the distribution $\mathcal{D}$. The query will be denoted Ex(). When the Oracle is only queried for a positive example we will write Pos-Ex(). Finally, if we pass a value m bounding the length of the admissible strings, we will write Ex(m). Combining ideas we can use Pos-Ex(m). The Oracle will return a string

drawn from $\mathcal{D}(\mathbb{L}(G))$ (for Pos-Ex()), $\mathcal{D}(\Sigma^{\leq m})$ (for Ex(m)) or $\mathcal{D}(\mathbb{L}(G) \cap \Sigma^{\leq m})$ (for Pos-Ex(m)), where we denote by $\mathcal{D}(L)$ the restriction of distribution $\mathcal{D}$ to the strings in L: $Pr_{\mathcal{D}(L)}(x) = \frac{Pr_{\mathcal{D}}(x)}{Pr_{\mathcal{D}}(L)}$ if $x \in L$, 0 if not. $Pr_{\mathcal{D}(L)}(x)$ is undefined if L is the empty set.

Definition 2 (Polynomial Pac-learnable). *Let $\mathcal{G}$ be a class of grammars. $\mathcal{G}$ is* Pac*-learnable if$_{def}$ there exists an algorithm **Alg** with the following property: For every grammar G in $\mathcal{G}$ of size at most n, for every distribution $\mathcal{D}$ over $\Sigma^\star$, for every $\epsilon > 0$ and $\delta > 0$, if **Alg** is given access to* Ex*(m), m and n, ϵ and δ then with probability at least 1-δ, **Alg** outputs an ϵ-good hypothesis with respect to G. If **Alg** runs in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, m and n, we say that $\mathcal{G}$ is* polynomiallly Pac-learnable.

Notice that in order to deal with the unbounded length of the examples we can use an $\epsilon' = \frac{\epsilon}{2}$ and a fraction of δ to compute m and accept to make an error of at most ϵ' over all the strings of length more than m, and then use Ex(m) instead of Ex().

We will denote by Poly$_{\text{Informant}}$-Pac the collection of all classes polynomially Pac-learnable.

We prove that provided $\mathcal{RP} \neq \mathcal{NP}$, good balls are not efficiently Pac-learnable. The proof follows the classical lines for such results: We first prove that the associated consistency problem is $\mathcal{NP}$-hard, through reductions from a well known $\mathcal{NP}$-complete problem (*Longest Common Subsequence*). Then it follows that if a polynomial Pac-learning algorithm for balls existed, this algorithm would provide us with a proof that this $\mathcal{NP}$-complete problem would also be in $\mathcal{RP}$.

Theorem 1. $\mathcal{GB}(\Sigma) \notin$ Poly$_{\text{Informant}}$-Pac .

Lemma 2. *The following problems are $\mathcal{NP}$-complete:*

Name: *Longest Common Subsequence (*Lcs*)*
Instance: *n strings $x_1 \dots x_n$, an integer k*
Question: *Does there exist a string w which is a subsequence of each x_i and is of length k?*

Name: *Longest Common Subsequence of Strings of a Given Length(*Lcssgl*)*
Instance: *n strings $x_1 \dots x_n$ all of length $2k$*
Question: *Does there exist a string w which is a subsequence of each x_i and is of length k?*

Name: *Consistent ball (*Cb*)*
Instance: *Two sets X_+ and X_- of strings over some alphabet Σ*
Question: *Does there exist a good ball containing X_+ and which does not intersect X_-?*

Proof (of lemmata). The first problem is solved in [22] (see also [18]). The second one can be found in [23] (Problem LCS0 (page 42)). For the last one, we use a reduction of problem LCSSGL: We take the strings of length $2k$, and put these with string λ into set X_+. Set X_- is constructed by taking each string of length $2k$ in X_+, inserting every possible symbol once only (hence constructing at most $n(2k+1)|\Sigma|$ strings of size $2k+1$). It follows that a ball that contains X_+ but no element of X_- has necessarily a centre of length k and a radius of k (since we focus on good balls). The centre is then a subsequence of all the strings of length $2k$ that were given. Conversely, if a ball is constructed using as centre a subsequence of length k, this ball is of radius k, contains also λ, and because of the radius, contains no element of X_-. Finally the problem is in $\mathcal{NP}$, since given a centre u it is easy to check if $\max_{x\in X_+} d(u,x) < \min_{x\in X_-} d(u,x)$. □

Proof (of Theo. 1). The proof relies on the widely accepted assumption that $\mathcal{NP} \neq \mathcal{RP}$, and follows the model introduced by Pitt and Valiant [21]. Suppose that $\boldsymbol{\mathcal{GB}}(\Sigma)$ is polynomially PAC-learnable with **Alg** and take an instance $\langle X_+, X_-\rangle$ of Problem CB. We write $h = |X_+| + |X_-|$ and define over $\Sigma^\star$ the distribution $Pr(x) = \frac{1}{h}$ if $x \in X_+ \cup X_-$, 0 if not. Let $\epsilon = \frac{1}{h+1}$, $\delta = \frac{1}{2}$, $m = n = \max\{|w| : w \in X_+\}$ and run $\mathbf{Alg}(\epsilon, \delta, m, n)$. Let $B_r(o)$ be the returned ball and test whether ($X_+ \subseteq B_r(o)$ and $X_- \cap B_r(o) = \emptyset$) or not. If there is no consistent ball, then $B_r(o)$ is necessarily inconsistent with the data, so the test above is false. If there is a consistent ball, then $B_r(o)$ is ϵ-good, with $\epsilon < \frac{1}{h}$. So, with probability at least $1-\delta > \frac{1}{2}$, there is no error at all and the test will be true. Finally, this procedure runs in polynomial time in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, m and n. Hence, if good balls were PAC-learnable, then there would be a randomized algorithm for the CB problem, proved $\mathcal{NP}$-complete by Lemma 2. □

3.2 About Pac-learning Balls from Positive Examples Only

In certain cases it may even be possible to PAC-learn from positive examples only. In this setting, during the learning phase, the examples are sampled following POS-EX() whereas during the testing phase the sampling is done following EX(), but in both cases the distribution is identical. Again, we can sample using POS-EX(m), where m is obtained by using Lemma 1 and little additional cost. Let us denote the collection of such classes polynomially PAC-learnable from TEXT by $\text{POLY}_{\text{TEXT}}$-PAC. Nevertheless, we get:

Theorem 2. $\boldsymbol{\mathcal{GB}}(\Sigma) \notin \text{POLY}_{\text{TEXT}}\text{-PAC}$.

Proof. Consider sample of strings $X_+ = \{a, b\}$. Given any ball B_1 containing X_+, there is another ball B_2 also containing a and b such that $B_1 - B_2 \neq \emptyset$; Let w_1 be a string in $B_1 - B_2$. Then we can construct a distribution $\mathcal{D}_1$ where $Pr_{\mathcal{D}_1}(a) = Pr_{\mathcal{D}_1}(b) = Pr_{\mathcal{D}_1}(w_1) = \frac{1}{3}$. But if from X_+ the Learner constructs B_1 instead of B_2, the error is of $\frac{1}{3}$ and cannot diminish as would be needed. □

4 Queries

Learning from queries involves the Learner (he) being able to interrogate the Oracle (she) using queries from a given set. The goal of the Learner is to identify the representation of an unknown language. The Oracle knows the target language and answers properly to the queries (*i.e.*, she does not lie). We call QUER the class of queries. For example, if the Learner is only allowed to make membership queries, we will have QUER = {MQ}.

Definition 3. *A class* $\mathcal{G}$ *is* polynomially identifiable in the limit with queries *from* QUER *if*$_{def}$ *there exists an algorithm* ***Alg*** *able to identify every* $G \in \mathcal{G}$ *such that, at any call of a query from* QUER*, the total number of queries and of time used up to that point by* ***Alg*** *is polynomial both in* $\|G\|$ *and in the size of the information presented up to that point by the Oracle.*

We will denote by POLY$_{\text{QUERIES}}$-QUER the collection of all classes polynomially identifiable in the limit with queries from QUER.

For instance, the class of all DFA is in POLY$_{\text{QUERIES}}$-{MQ, EQ} [24]. In the case of good balls, we get the following result:

Theorem 3. $\mathcal{GB}(\Sigma) \notin$POLY$_{\text{QUER}}$-{MQ, EQ}.

Proof. Let $n \in \mathbb{N}$ and $\mathcal{B}_{\leq n} = \{B_r(o) : o \in \Sigma^\star, r \leq |o| \leq n\}$. Following [6], we describe an Adversary who maintains a set S of all possible balls. At the beginning, $S = \mathcal{B}_{\leq n}$. Her answer to the equivalence query $L = B_r(o)$ is a counterexample o. Her answer to the membership query o is NO. At each step, the Adversary eliminates many balls of S but only one of centre o and radius 0. As there are $\Omega(|\Sigma|^n)$ such balls in $\mathcal{B}_{\leq n}$, identifying them requires $\Omega(|\Sigma|^n)$ queries. □

Notice however that if the Learner is given one string from a good ball, then he can learn using a polynomial number of MQ only. Also, we have shown in [15] that special queries, called *correction queries* (CQ), allowed to identify $\mathcal{GB}(\Sigma)$. Given a language L, a correction of a string w is either YES if $w \in L$, or a string $w' \in L$ at minimum edit distance from w, if $w \notin L$.

Theorem ([15]). $\mathcal{GB}(\Sigma) \in$POLY$_{\text{QUER}}$-{CQ}.

5 Polynomial Identification of Balls

In Gold's standard identification in the limit paradigm, a Learner receives an infinite sequence of information (presentation) that should help him to find the representation $G \in \mathcal{G}$ of an unkown target language L. The set of admissible presentations is denoted by **Pres**, each presentation being a function $\mathbb{N} \to X$ where X is any set. Given $\boldsymbol{f} \in$ **Pres**, we will denote by $\boldsymbol{f}_m$ the $m+1$ first elements of $\boldsymbol{f}$, and by $\boldsymbol{f}(n)$ its n^{th} element. Below, we will concentrate on two sorts of presentations:

– **Pres**=TEXT: All the strings in L are presented: $\boldsymbol{f}(\mathbb{N}) = \mathbb{L}(G)$
– **Pres**=INFORMANT: The presentation is of labelled pairs (w, l) where $(w \in L \implies l = +)$ and $(w \notin L \implies l = -)$: $\boldsymbol{f}(\mathbb{N}) = \mathbb{L}(G) \times \{+\} \cup \overline{\mathbb{L}(G)} \times \{-\}$.

Definition 4. *We say that $\boldsymbol{\mathcal{G}}$ is identifiable in the limit from **Pres** if$_{def}$ there exists a learning algorithm **Alg** such that for all $G \in \boldsymbol{\mathcal{G}}$ and for any presentation $\boldsymbol{f}$ of $\mathbb{L}(G)$ (belonging to **Pres**), there exists a rank n such that for all $m \geq n$, $\mathbb{L}(\boldsymbol{Alg}(\boldsymbol{f}_m)) = \mathbb{L}(G)$.*

This definition yields a lot of learnability results. However, the absence of efficiency constraints may lead to unusable algorithms. Therefore several authors have tried to define *polynomial* identification in the limit, by introducing different efficiency criteria and combining them.

5.1 Polynomial Identification Criteria

Firstly, it is reasonable to think that polynomiality must concern the amount of time an algorithm has to learn:

Definition 5 (Polynomial Update Time). *An algorithm **Alg** is said to have* polynomial update time *if$_{def}$ there is a polynomial $p()$ such that, for every presentation $\boldsymbol{f}$ and every integer n, constructing $\boldsymbol{Alg}(\boldsymbol{f}_n)$ requires $\mathcal{O}(p(\|\boldsymbol{f}_n\|))$ time.*

However, it is known that polynomial update time is not sufficient [7]. Indeed, a Learner could receive an exponential number of examples without doing anything but wait, and then use the huge amount of time he saved in order to solve any $\mathcal{NP}$-hard problem...

Secondly, polynomiality should also concern the minimum amount of data that any algorithm should receive in order to learn:

Definition 6 (Polynomial Characteristic Sample). *A grammar class $\boldsymbol{\mathcal{G}}$ admits* polynomial characteristic samples *if$_{def}$ there exist an algorithm **Alg** and a polynomial $p()$ such that $\forall G \in \boldsymbol{\mathcal{G}}$ $\exists$Cs $\subseteq X$ $\forall \boldsymbol{f} \in \boldsymbol{Pres}$ $\forall n \in \mathbb{N} : \|\text{Cs}\| \leq p(\|G\|) \wedge \text{Cs} \subseteq \boldsymbol{f}_n \implies \mathbb{L}(\boldsymbol{Alg}(\boldsymbol{f}_n)) = \mathbb{L}(G)$. Such a set* Cs *is called a* characteristic sample *of G for **Alg**. If such an algorithm **Alg** exists, we say that* **Alg** identifies $\boldsymbol{\mathcal{G}}$ in the limit in POLY-CS time for **Pres**.

> We will denote by POLY$_{\text{PRESENTATION}}$-CS the collection of all classes identifiable in the limit in POLY-CS time from PRESENTATION (either TEXT or INFORMANT).

Notice that if a grammar class only admits characteristic samples whose size are exponential, then no algorithm will be able to converge before receiving an unreasonable amount of data. So the existence of polynomial characteristic sample is necessary but not sufficient again. Notice that several authors (*e.g.*, [11]) have used stronger notions of polynomial characteristic samples to define polynomial identification in the limit.

Thirdly, polynomiality can concern the behaviour of the algorithm itself through the hypotheses he outputs all along his learning, *e.g.*, the number of implicit prediction errors [7]:

Definition 7 (Implicit Prediction Errors). *Given a learning algorithm* $\boldsymbol{Alg}$ *and a presentation* $\boldsymbol{f}$*, we say that* $\boldsymbol{Alg}$ makes an implicit prediction error at time n *if*$_{def}$ $\boldsymbol{Alg}(\boldsymbol{f}_{n-1}) \nvdash \boldsymbol{f}(n)$*. We say that* $\boldsymbol{Alg}$ *is* consistent *if*$_{def}$ *it changes its mind each time a prediction error is detected with the new presented element.*

Definition 8 (Polynomial Implicit Prediction Errors Criterion). *An algorithm* $\boldsymbol{Alg}$ *identifies a class* $\boldsymbol{\mathcal{G}}$ *in the limit in* Ipe *polynomial time if*$_{def}$ *(1)* $\boldsymbol{Alg}$ *identifies* $\boldsymbol{\mathcal{G}}$ *in the limit, (2)* $\boldsymbol{Alg}$ *has polynomial update time and (3)* $\boldsymbol{Alg}$ *makes a polynomial number of implicit prediction errors: Let* $\#\text{Ipe}(\boldsymbol{f}) = |\{k \in \mathbb{N} : \boldsymbol{Alg}(\boldsymbol{f}_k) \nvdash \boldsymbol{f}(k+1)\}|$*; There exists a polynomial* $p()$ *such that, for each grammar* G *and each presentation* $\boldsymbol{f}$ *of* $\mathbb{L}(G)$*,* $\#\text{Ipe}(\boldsymbol{f}) \leq p(\|G\|)$*.*

Note that the first condition is not implied by the two others.

We will denote by $\text{Poly}_{\text{Presentation}}$-Ipe the collection of all classes polynomially identifiable in the limit in Ipe polynomial time from Presentation (either Text or Informant).

Fourthly, one can bound the number of mind changes instead of Ipe [25].

Definition 9 (Mind Changes). *Given a learning algorithm* $\boldsymbol{Alg}$ *and a presentation* $\boldsymbol{f}$*, we say that* $\boldsymbol{Alg}$ changes its mind at time n *if*$_{def}$ $\boldsymbol{Alg}(\boldsymbol{f}_n) \neq \boldsymbol{Alg}(\boldsymbol{f}_{n-1})$*. We say that* $\boldsymbol{Alg}$ *is* conservative *if*$_{def}$ *it never changes its mind when the current hypothesis is consistent with the new presented element.*

Definition 10 (Polynomial Mind Changes Criterion). *An algorithm* $\boldsymbol{Alg}$ *identifies a class* $\boldsymbol{\mathcal{G}}$ *in the limit in* Mc *polynomial time if*$_{def}$ *(1)* $\boldsymbol{Alg}$ *identifies* $\boldsymbol{\mathcal{G}}$ *in the limit, (2)* $\boldsymbol{Alg}$ *has polynomial update time and (3)*$\boldsymbol{Alg}$ *makes a polynomial number of mind changes: Let* $\#\text{Mc}(\boldsymbol{f}) = |\{k \in \mathbb{N} : \boldsymbol{Alg}(\boldsymbol{f}_k) \neq \boldsymbol{Alg}(\boldsymbol{f}_{k+1})\}|$*; There exists a polynomial* $p()$ *such that, for each grammar* G *and each presentation* $\boldsymbol{f}$ *of* $\mathbb{L}(G)$*,* $\#\text{Mc}(\boldsymbol{f}) \leq p(\|G\|)$*.*

We will denote by $\text{Poly}_{\text{Presentation}}$-Mc the collection of all classes polynomially identifiable in the limit in Mc polynomial time from Presentation (either Text or Informant).

Finally, if an algorithm **Alg** is consistent then $\#\text{Ipe}(\boldsymbol{f}) \leq \#\text{Mc}(\boldsymbol{f})$ for every presentation $\boldsymbol{f}$. Likewise, if **Alg** is conservative then $\#\text{Mc}(\boldsymbol{f}) \leq \#\text{Ipe}(\boldsymbol{f})$. So we deduce the following theorem:

Theorem 4. *If* $\boldsymbol{Alg}$ *identifies the class* $\boldsymbol{\mathcal{G}}$ *in the limit in* Mc *polynomial time and is consistent, then* $\boldsymbol{Alg}$ *identifies* $\boldsymbol{\mathcal{G}}$ *in the limit in* Ipe *polynomial time. Conversely, if* $\boldsymbol{Alg}$ *identifies* $\boldsymbol{\mathcal{G}}$ *in the limit in* Ipe *polynomial time and is conservative, then* $\boldsymbol{Alg}$ *identifies* $\boldsymbol{\mathcal{G}}$ *in the limit in* Mc *polynomial time.*

5.2 Identification Results for Texts

In this section, we show the following results:

Theorem 5. *1.* $\mathcal{GB}(\Sigma) \in$ Poly$_{\text{Text}}$-Ipe*;*
2. $\mathcal{GB}(\Sigma) \in$ Poly$_{\text{Text}}$-Mc*;*
3. $\mathcal{GB}(\Sigma) \in$ Poly$_{\text{Text}}$-Cs.

We say that u is a *subsequence* of v, denoted $u \preceq v$, *if*$_{def}$ $u = a_1 \dots a_n$ and there exist $u_0, \dots, u_n \in \Sigma^\star$ such that $v = u_0 a_1 u_1 \dots a_n u_n$. Subsequences and edit distance are strongly related since, $d(w, w') \geq \big||w| - |w'|\big|$. Moreover, $d(w, w') = \big||w| - |w'|\big|$ iff ($w \preceq w'$ or $w' \preceq w$). We denote by $lcs(u, v)$ the set of longest common subsequences of u and v.

In order to prove Theo. 5, we will need to build the minimum consistent ball containing a set $S = \{x_1, \dots x_n\}$ of strings (learning sample). This will be efficiently achievable if S admits a so-called *certificate*. We denote by $S^{max} = \{w \in S : \forall u \in S, |w| \geq |u|\}$ the set of strings in S of maximum length and by $S^{min} = \{w \in S : \forall u \in S, |w| \leq |u|\}$, the set of strings in S of minimum length.

Definition 11 (Certificate). *A certificate for S is a tuple (u, v, w, o, r) such that (1) $u, v \in S^{max}$, (2) $w \in S^{min}$, (3) $|u| - |w| = 2r (= |v| - |w|)$, (4) $lcs(u, v) = \{o\}$, (5) $|o| = |u| - r (= |v| - r)$ and (6) $S \subseteq B_r(o)$. There may exist 0, 1 or several certificates for S; We will say that S* admits *a certificate if*$_{def}$ *there is at least one.*

Lemma 3. *Suppose that S admits the certificate (u, v, w, o, r). If $S \subseteq B_{r'}(o')$ then either $r' > r$, or ($r' = r$ and $o' = o$). In other words, $B_r(o)$ is the smallest ball containing S (w.r.t. the radius). Moreover, if (u, v, w, o, r) and (u', v', w', o', r') are 2 certificates for S, then ($o = o'$ and $r = r'$).*

Proof. We have $d(u, w) \geq |u| - |w| = 2r$. Moreover, as $\{u, w\} \subset S \subseteq B_{r'}(o')$, we deduce that $d(u, w) \leq 2r'$ (since $2r'$ is the diameter of $B_{r'}(o')$). So $r \leq r'$. If $r < r'$, the result is immediate. Suppose that $r' = r$. Then $d(u, w) = 2r \leq d(u, o') + d(o', w)$. As $\{u, w\} \subset S \subseteq B_{r'}(o')$, we get $d(u, o') \leq r$ and $d(o', w) \leq r$, thus $d(u, o') = d(o', w) = r$. As $|u| - |o'| \leq d(u, o') = r$, we get $|o'| \geq |u| - r = |o|$. Conversely, $|o'| - |w| \leq d(o', w) = r$, so $|o'| \leq |w| + r = |u| - 2r + r = |o|$. So $|o'| = |o|$. If $o' \neq o$, as $lcs(u, v) = \{o\}$ and $|o| = |o'|$, we deduce that $o' \notin lcs(u, v)$, so either $o' \not\preceq u$ or $o' \not\preceq v$. Assume $o' \not\preceq u$ *w.l.o.g.*. Then, $d(o', u) > |u| - |o'| = r$, which is impossible since $u \in B_{r'}(o')$. Hence, $o' = o$. Finally, if (u, v, w, o, r) and (u', v', w', o', r') are 2 certificates for S, then $S \subseteq B_r(o)$ and $S \subseteq B_{r'}(o')$, so ($o = o'$ and $r = r'$). □

Lemma 4. *There is a polynomial algorithm (in $\|S\|$) that (1) checks whether S admits a certificate, and if so, (2) exhibit one of them.*

Proof. If, for any $u \in S^{max}$ and $w \in S^{min}$, $|u| - |w|$ is odd, then there is no certificate, else, let $r = (|u| - |w|)/2$. The next step is to find 2 strings $u, v \in S^{max}$ for which $\exists o \in \Sigma^\star$ such that (4) $lcs(u, v) = \{o\}$, (5) $|o| = |u| - r$

and (6) $S \subseteq B_r(o)$. Checking if (5) and (6) hold is polynomial as soon as one gets (4) holds. Nevertheless, the number of distinct strings in $lcs(u, v)$ can be as large as approximately 1.442^n [26] (where $n = |u| = |v|$). So it is inconceivable to enumerate $lcs(u, v)$ with a brute-force algorithm. On the other hand, several papers overcome this problem by producing compact representations of $lcs(u, v)$. In [27], the so-called Lcs-*graph* can be computed in $\mathcal{O}(n^2)$ time and it is easy to check if it contains exactly 1 string. Hence, checking if a certificate exists for S and computing it can be acheived in at most $\mathcal{O}(\|S\|^4)$ time. □

Algorithm 1: Pseudo Algorithm for the Identification of Good Balls from Text.

```
Data: A text f = {x_1, x_2, ...}
read x_1;
output (x_1, 0) ;
c ← x_1;
while true do
    read x_i;
    if there exists a certificate (u, v, w, o, r) for f_i then
        output (o, r) (* valid ball *)
    else
        if c ∉ S^max then c ← one string of S^max;
        output (c, |c|) (* junk ball *)
    end
end
```

Lemma 5. *Algo. 1 identifies* $\mathcal{GB}(\Sigma)$ *in the limit.*

Proof. Assume that $B_r(o)$ is the target ball. It is easy to see that there exists a lot of certificates (u, v, w, o, r) for $B_r(o)$. *E.g.*, $u = a^r o, v = b^r o$ and w is any string of $B_r(o)$ of length $|o| - r$ (where $a, b \in \Sigma, a \neq b$). As soon as such u, v, w appears in $\boldsymbol{f}_i$, then for all $j \geq i$ (u, v, w, o, r) will be a certificate for $\boldsymbol{f}_j$. Note that other certificates may exist, but due to Lemma 3, Algo. 1 will return the same representation (o, r) forever. □

Lemma 6. *Algo. 1 makes a polynomial number of* Mc.

Proof. Assume that $B_r(o)$ is the target ball and $\boldsymbol{f}$ is a presentation. Let us run Algo. 1 on $\boldsymbol{f}$ and observe the output trace $T = (x_1, s_1)(x_2, s_2)(x_3, s_3)\ldots$ Each (x_i, s_i) is either the representation of a valid ball (o, r) coming from a certificate, or else, of a junk ball $(c, |c|)$.

Let (o_i, r_i) be an output generated by a certificate and j the smallest rank such that $j > i$ and (o_j, r_j) is also valid. If $\boldsymbol{f}(i+1) \in B_{r_i}(o_i)$ then $(u_i, v_i, w_i, o_i, r_i)$ is still a certificate for $\boldsymbol{f}_{i+1}$ (by Def. 11), so by Lemma 3, $j = i+1$ and $(o_j, r_j) = (o_i, r_i)$. Else, $\boldsymbol{f}(i+1) \notin B_{r_i}(o_i)$. Then, as $\boldsymbol{f}_i \subseteq B_{r_j}(o_j)$, by Lemma 3, either $r_i < r_j$ or $(o_j = o_i$ and $r_j = r_i)$. The latter is impossible, so $r_i < r_j$. Therefore, each

time **Alg** changes its mind in favor of a new valid ball, its radius is increased by at least 1 *w.r.t.* the previous valid balls. So the number of different valid balls it will output will be at most r (it is bounded by the radius of the target ball). Moreover, the number of Mc of **Alg** *in favor of* a valid ball is $\leq r$.

On the other hand, let $(c_i, |c_i|)$ and $(c_j, |c_j|)$ be 2 junk balls. We have: (1) if $i < j$ then $|c_i| \leq |c_j|$ (since c_i (*resp.* c_j) is a string of maximum length in $\boldsymbol{f}_i$ (*resp.* $\boldsymbol{f}_j$)), and (2) if $|c_i| = |c_j|$ then $c_i = c_j$. So the number of different junk balls all along T is bounded by $2r$ (since $\forall x \in B_r(o), |o| - r \leq |x| \leq |o| + r$). Moreover, the number of Mc *in favor of* a junk ball is $\leq 3r$, that is, r Mc to change from a valid ball to a junk ball and $2r$ Mc to change from a junk ball to a junk ball. Hence the total amount of Mc is $\leq 4r$. □

Proof (of Theo. 5). (2) Algo. 1 identifies in the limit in Mc polynomial time from text as a consequence of Lemmata 4, 5 and 6. (1) Algo. 1 is clearly consistent. Indeed, either one gets (o, r) that comes from some certificate, and then $\boldsymbol{f}_i \subseteq B_r(o)$; Or one gets $(c, |c|)$ with $c \in S^{max}$, and then $\boldsymbol{f}_i \subseteq \Sigma^{\leq |c|} \subseteq B_{|c|}(c)$. As $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Text}}$-Mc, we get $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Text}}$-Ipe by Theo. 4. (3) Any certificate of the target ball is the base of a characteristic sample that makes Algo. 1 converge towards the target, so $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Text}}$-Cs. □

5.3 Identification Results for Informants

Theorem 6. *1.* $\boldsymbol{\mathcal{GB}}(\Sigma) \notin$ Poly$_{\text{Informant}}$-Ipe*;*
2. $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Informant}}$-Mc*;*
3. $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Informant}}$-Cs*.*

Proof. (1) Following [7], we suppose that $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Informant}}$-Ipe. Then, $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Queries}}$-{Eq}, that contradicts Theo. 3. Indeed, let **Alg** be an algorithm that would yield $\boldsymbol{\mathcal{GB}}(\Sigma) \in$ Poly$_{\text{Informant}}$-Ipe and consider an Oracle that is able to answer to Eq. Assume that the Learner's current hypothesis is the ball B. He submits it to the Oracle. Either he gets Yes and the inference task ends. Or he gets a counterexample x and uses **Alg** on x to produce a new hypothesis B'. As the counterexamples provided by the Oracle are prediction errors *w.r.t.* the hypotheses provided by **Alg**, we deduce that if **Alg** makes a polynomial number of Ipe, then the Learner identifies $\boldsymbol{\mathcal{GB}}(\Sigma)$ with a polynomial number of Eq. (2) As hypotheses do not have to be consistent with the data, we can use Algo. 1 to identify from an informant by ignoring the negative examples. (3) We use the same characteristic samples as in the proof of Theo. 5 (3). □

6 Results for $\boldsymbol{\mathcal{BALL}}(\Sigma)$

By simple corollaries of the above theorems, we have the following negatives results:

Theorem 7. *1.* $\boldsymbol{\mathcal{BALL}}(\Sigma) \notin$ Poly$_{\text{Presentation}}$-Pac*;*
2. $\boldsymbol{\mathcal{BALL}}(\Sigma) \notin$ Poly$_{\text{Queries}}$-{Mq, Eq}*;*

3. $\boldsymbol{\mathcal{BALL}}(\Sigma) \notin \text{POLY}_{\text{QUERIES}}\text{-}\{\text{CQ}\}$;
4. $\boldsymbol{\mathcal{BALL}}(\Sigma) \notin \text{POLY}_{\text{INFORMANT}}\text{-IPE}$;
5. $\boldsymbol{\mathcal{BALL}}(\Sigma) \notin \text{POLY}_{\text{PRESENTATION}}\text{-CS}$.

Proof. (1), (2), (4) come from our results on $\boldsymbol{\mathcal{GB}}(\Sigma)$. (3) One cannot learn $B_n(\lambda)$ without making a query outside the ball, that would involve a string of exponential length. (5) Characteristic samples for $B_n(\lambda)$ and $B_{n+1}(\lambda)$ cannot be different over strings of polynomial (in $\log n$) length. □

Theorem 8. 1. $\boldsymbol{\mathcal{BALL}}(\Sigma) \notin \text{POLY}_{\text{TEXT}}\text{-IPE}$;
2. $\boldsymbol{\mathcal{BALL}}(\Sigma) \notin \text{POLY}_{\text{TEXT}}\text{-MC}$.

Proof. We give the proof here for the case where the learning algorithm is deterministic. If the algorithm is allowed to be randomized, a slightly more tedious proof can be derived from this one. Suppose we have a Learner **Alg** and a polynomial $p()$ such that $\forall G \in \boldsymbol{\mathcal{G}}, \forall \boldsymbol{f} \in \textbf{Pres}, \#\text{MC}(\boldsymbol{f}) \leq p(\|G\|)$. Let n be a sufficiently large integer, and consider the subclass of targets $B_k(\lambda)$ with $k \leq n$. For each target, we construct a presentation $\boldsymbol{f}^k$ by running **Alg** in an interactive learning session. At each step i, **Alg** produces hypothesis H_i, and we have to compute a new string $\boldsymbol{f}^k(i+1)$. For this purpose, we use Algo. 2.

Algorithm 2: Compute $\boldsymbol{f}^k(i)$

Data: i, $\boldsymbol{f}^k_{i-1}$, $H_0 \dots H_{i-1}$
Result: $\boldsymbol{f}^k(i)$
if $i = 0$ **then return** λ
else
  if $H_{i-1} = B_k(\lambda)$ **then**
    if $B_k(\lambda) - \boldsymbol{f}^k_{i-1} \neq \emptyset$ **then return** $\min(B_k(\lambda) - \boldsymbol{f}^k_{i-1})$
    else return λ
  else
    if $H_{i-1} = B_j(\lambda)$ *where* $j = \max\{|u| : u \in \boldsymbol{f}^k_{i-1}\}$ **then return** a^{j+1}
    else return λ
  end
end

Each presentation $\boldsymbol{f}^k$ is a correct text presentation of its target $B_k(\lambda)$, *i.e.*, $\boldsymbol{f}^k(\mathbb{N}) = B_k(\lambda)$. Let us denote by $m(k) = \min\{i \in \mathbb{N} : \boldsymbol{f}^k(i) = a^k\}$. For each k, $\boldsymbol{f}^k$ and $\boldsymbol{f}^{k+1}$ coincide on the same $m(k)$ initial values. Then $\boldsymbol{f}^n$ can be rewritten as: $\lambda, \dots, \lambda, a, \dots, a, \dots, a^n, \dots$ with: $\forall j : 0 < j \leq n, \forall i \in \{m(j-1), .., m(j) - 1\}, \boldsymbol{f}^n(i) = \boldsymbol{f}^j(m(j-1)) = \boldsymbol{f}^j(i) = a^{j-1}$, and **Alg** makes a mind change just before receiving new example a^i. This proves that $\#\text{MC}(\boldsymbol{f}^n) > p(\log n)$. Therefore, given any learning algorithm **Alg** and any polynomial $p()$, there is a $n \in \mathbb{N}$ such that $\#\text{MC}(\boldsymbol{f}^n) > p(\log n)$. With similar arguments, we get $\#\text{IPE}(\boldsymbol{f}^n) > p(\log n)$. □

Theorem 9. $\boldsymbol{\mathcal{BALL}}(\Sigma) \in \text{POLY}_{\text{INFORMANT}}\text{-MC}$

Proof. We prove that there is a Learner that just checks the data until it is sure that there is only one possible consistent ball and therefore makes just one mind change. Let $B_r(o)$ be the target ball, and $\langle X_+, X_- \rangle$ be a sample such that there is a string u for which (1)$a^k u, b^k u \in X_+$, (2) all supersequences of $a^k u$ or of $b^k u$ of length $|u|+1+k$ are in X_- and (3) if $u \neq \lambda$, for each subsequence v of u of length $|u|-1$, there is a supersequence of v of length $|u|+k$ in X_-. Note that (1) given a ball $B_r(o)$, this sample always exists; (2) Checking if there is such a string u for a sample X is in $\mathcal{O}(\|X\|)$; (3) All edit operations in a minimal path transforming o into $a^k u$ and $b^k u$ are insertions: If not, by changing any non-insertion operation by an insertion, we can build a string w such that $d(o, w) = d(o, a^k u) \wedge w \in X_-$; Therefore $o \preceq u$; (4) Since for each subsequence w of u there is a supersequence of length $|u| + k$ in X_-, no proper subsequence of u is the centre. We conclude that $u = o$ and $k = r$. And of course the required conditions will be true at some point of the presentation. □

7 Conclusion

Criterion	$\mathcal{GB}(\Sigma)$		$\mathcal{BALL}(\Sigma)$	
Poly$_{\text{Informant}}$-Pac	No	Theo. 1	No	Theo. 7 (1)
Poly$_{\text{Text}}$-Pac	No	Theo. 2	No	Theo. 7 (1)
Poly$_{\text{Informant}}$-Ipe	No	Theo. 6 (1)	No	Theo. 7 (4)
Poly$_{\text{Text}}$-Ipe	Yes	Theo. 5 (1)	No	Theo. 8 (1)
Poly$_{\text{Informant}}$-Mc	Yes	Theo. 6 (2)	Yes	Theo. 9
Poly$_{\text{Text}}$-Mc	Yes	Theo. 5 (2)	No	Theo. 8 (2)
Poly$_{\text{Informant}}$-Cs	Yes	Theo. 6 (3)	No	Theo. 7 (5)
Poly$_{\text{Text}}$-Cs	Yes	Theo. 5 (3)	No	Theo. 7 (5)
Poly$_{\text{Queries}}$-{Mq, Eq}	No	Theo. 3	No	Theo. 7 (2)
Poly$_{\text{Queries}}$-{Cq}	Yes	Theo. ([15])	No	Theo. 7 (3)

An alternative to making a difference between $\mathcal{BALL}(\Sigma)$ and $\mathcal{GB}(\Sigma)$ is to consider an intermediary collection of classes: For any polynomial $p()$, $p()$-good balls are those balls $B_r(o)$ for which $r \leq p(|o|)$, and we denote by $p() - \mathcal{GB}(\Sigma)$ the corresponding class. It seems that if most results for good balls transfer to $p()$-good balls in a natural way, this is not systematically the case.

References

1. Gold, E.M.: Language identification in the limit. Information and Control **10**(5) (1967) 447–474
2. Gold, E.M.: Complexity of automaton identification from given data. Information and Control **37** (1978) 302–320
3. Valiant, L.G.: A theory of the learnable. Communications of the Association for Computing Machinery **27**(11) (1984) 1134–1142

4. Angluin, D.: Queries and concept learning. Machine Learning Journal **2** (1987) 319–342
5. Jain, S., Osherson, D., Royer, J.S., Sharma, A.: Systems That Learn. MIT press (1999)
6. Angluin, D.: Negative results for equivalence queries. Machine Learning Journal **5** (1990) 121–150
7. Pitt, L.: Inductive inference, DFA's, and computational complexity. In: Analogical and Inductive Inference. Number 397 in LNAI. Springer-Verlag (1989) 18–44
8. Li, M., Vitanyi, P.: Learning simple concepts under simple distributions. Siam J. of Comput. **20** (1991) 911–935
9. Parekh, R.J., Honavar, V.: On the relationship between models for learning in helpful environments. In: Proc. of Grammatical Inference: Algorithms and Applications (ICGI'00), LNAI 1891 (2000) 207–220
10. Kearns, M., Valiant, L.: Cryptographic limitations on learning boolean formulae and finite automata. In: 21st ACM Symposium on Theory of Computing. (1989) 433–444
11. de la Higuera, C.: Characteristic sets for polynomial grammatical inference. Machine Learning Journal **27** (1997) 125–138
12. Zeugmann, T.: Can learning in the limit be done efficiently? In: Proc. of Algorithmic Learning Theory (ALT'03), LNCS 2842 (2003) 17–38
13. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Doklady Akademii Nauk SSSR **163**(4) (1965) 845–848
14. Tantini, F., de la Higuera, C., Janodet, J.C.: Identification in the limit of systematic-noisy languages. In: Proc. of Grammatical Inference: Algorithms and Applications (ICGI'06), LNAI 4201 (2006) 19–31
15. Becerra-Bonache, L., de la Higuera, C., Janodet, J.C., Tantini, F.: Learning balls of strings with correction queries. Technical report, University of Saint-Etienne, http://labh-curien.univ-st-etienne.fr/ tantini/pub/bhjt07Long.pdf (2007)
16. Wagner, R., Fisher, M.: The string-to-string correction problem. Journal of the ACM **21** (1974) 168–178
17. Crochemore, M., Hancart, C., Lecroq, T.: Algorithmique du Texte. Vuibert (2001)
18. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman (1979)
19. Warmuth, M.: Towards representation independence in *pac*-learning. In: Proc. International Workshop on Analogical and Inductive Inference (AII'89), LNAI 397 (1989) 78–103
20. Kearns, M., Vazirani, U.: An Introduction to Computational Learning Theory. MIT press (1994)
21. Pitt, L., Valiant, L.G.: Computational limitations on learning from examples. Journal of the ACM **35**(4) (1988) 965–984
22. Maier, D.: The complexity of some problems on subsequences and supersequences. Journal of the ACM **25** (1977) 322–336
23. de la Higuera, C., Casacuberta, F.: Topology of strings: Median string is NP-complete. Theoretical Computer Science **230** (2000) 39–48
24. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Control **39** (1987) 337–350
25. Angluin, D., Smith, C.: Inductive inference: theory and methods. ACM computing surveys **15**(3) (1983) 237–269
26. Greenberg, R.I.: Bounds on the number of longest common subsequences. Technical report, Loyola University (2003)
27. Greenberg, R.I.: Fast and simple computation of all longest common subsequences. Technical report, Loyola University (2002)